**Задание:**

Для каждого дня в таблице orders рассчитайте следующие показатели:

1. Выручку, полученную в этот день.
2. Суммарную выручку на текущий день.
3. Прирост выручки, полученной в этот день, относительно значения выручки за предыдущий день.

Колонки с показателями назовите соответственно revenue, total\_revenue, revenue\_change. Колонку с датами назовите date.

Прирост выручки рассчитайте в процентах и округлите значения до двух знаков после запятой.

Результат должен быть отсортирован по возрастанию даты.

Поля в результирующей таблице: date, revenue, total\_revenue, revenue\_change

**Пояснение:**

Будем считать, что оплата за заказ поступает сразу же после его оформления, т.е. случаи, когда заказ был оформлен в один день, а оплата получена на следующий, возникнуть не могут.

Суммарная выручка на текущий день — это результат сложения выручки, полученной в текущий день, со значениями аналогичного показателя всех предыдущих дней.

При расчёте выручки помните, что не все заказы были оплачены — некоторые были отменены пользователями.

Не забывайте при делении заранее приводить значения к нужному типу данных. Пропущенные значения прироста для самой первой даты не заполняйте — просто оставьте поля в этой строке пустыми.

with revenue\_by\_date as (SELECT date,

sum(price) as revenue

FROM (SELECT creation\_time::date as date,

order\_id,

unnest(product\_ids) as product\_id

FROM orders) t join products using(product\_id)

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')

GROUP BY date

ORDER BY date)

SELECT date,

revenue,

sum(revenue) OVER(ORDER BY date) as total\_revenue,

round((revenue - lag(revenue, 1) OVER())::decimal / lag(revenue, 1) OVER() \* 100,

2) as revenue\_change

FROM revenue\_by\_date

Теперь на основе данных о выручке рассчитаем несколько относительных показателей, которые покажут, сколько в среднем потребители готовы платить за услуги нашего сервиса доставки. Остановимся на следующих метриках:

1. ARPU (Average Revenue Per User) — средняя выручка на одного пользователя за определённый период.

2. ARPPU (Average Revenue Per Paying User) — средняя выручка на одного платящего пользователя за определённый период.

3. AOV (Average Order Value) — средний чек, или отношение выручки за определённый период к общему количеству заказов за это же время.

Если за рассматриваемый период сервис заработал 100 000 рублей и при этом им пользовались 500 уникальных пользователей, из которых 400 сделали в общей сложности 650 заказов, тогда метрики будут иметь следующие значения:

*ARPU* =100000/500=200

*ARPPU* =100000/400=250

*AOV*=100000/650≈153,85

**Задание:**

Для каждого дня в таблицах orders и user\_actions рассчитайте следующие показатели:

1. Выручку на пользователя (ARPU) за текущий день.
2. Выручку на платящего пользователя (ARPPU) за текущий день.
3. Выручку с заказа, или средний чек (AOV) за текущий день.

Колонки с показателями назовите соответственно arpu, arppu, aov. Колонку с датами назовите date.

При расчёте всех показателей округляйте значения до двух знаков после запятой.

Результат должен быть отсортирован по возрастанию даты.

Поля в результирующей таблице: date, arpu, arppu, aov

**Пояснение:**

Будем считать, что оплата за заказ поступает сразу же после его оформления, т.е. случаи, когда заказ был оформлен в один день, а оплата получена на следующий, возникнуть не могут.

Платящими будем считать тех пользователей, которые в данный день оформили хотя бы один заказ, который в дальнейшем не был отменен.

SELECT date,

round(sum(price)::decimal / all\_users, 2) as arpu,

round(sum(price)::decimal / count(distinct user\_id), 2) as arppu,

round(sum(price)::decimal / count(distinct order\_id), 2) as aov

FROM (SELECT time::date as date,

order\_id,

user\_id

FROM user\_actions

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')) t join (SELECT order\_id,

unnest(product\_ids) as product\_id

FROM orders) t2 using(order\_id) join products using(product\_id) join (SELECT time::date as date,

count(distinct user\_id) as all\_users

FROM user\_actions

GROUP BY time::date) t3 using(date)

GROUP BY date, all\_users;

Давайте посчитаем те же показатели, но в другом разрезе — не просто по дням, а по дням недели.

**Задание:**

Для каждого дня недели в таблицах orders и user\_actions рассчитайте следующие показатели:

1. Выручку на пользователя (ARPU).
2. Выручку на платящего пользователя (ARPPU).
3. Выручку на заказ (AOV).

При расчётах учитывайте данные только за период с 26 августа 2022 года по 8 сентября 2022 года включительно — так, чтобы в анализ попало одинаковое количество всех дней недели (ровно по два дня).

В результирующую таблицу включите как наименования дней недели (например, Monday), так и порядковый номер дня недели (от 1 до 7, где 1 — это Monday, 7 — это Sunday).

Колонки с показателями назовите соответственно arpu, arppu, aov. Колонку с наименованием дня недели назовите weekday, а колонку с порядковым номером дня недели weekday\_number.

При расчёте всех показателей округляйте значения до двух знаков после запятой.

Результат должен быть отсортирован по возрастанию порядкового номера дня недели.

Поля в результирующей таблице:

weekday, weekday\_number, arpu, arppu, aov

**Пояснение:**

Будем считать, что оплата за заказ поступает сразу же после его оформления, т.е. случаи, когда заказ был оформлен в один день, а оплата получена на следующий, возникнуть не могут.

Платящими будем считать тех пользователей, которые в данный день оформили хотя бы один заказ, который в дальнейшем не был отменен.

SELECT concat(upper(left(weekday, 1)), lower(substring(weekday, 2))) as weekday,

weekday\_number,

round(sum(price)::decimal / all\_users, 2) as arpu,

round(sum(price)::decimal / count(distinct user\_id), 2) as arppu,

round(sum(price)::decimal / count(distinct order\_id), 2) as aov

FROM (SELECT to\_char(time, 'day') as weekday,

date\_part('isodow', time) as weekday\_number,

order\_id,

user\_id

FROM user\_actions

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')

and time::date between '2022-08-26'

and '2022-09-08') t join (SELECT order\_id,

unnest(product\_ids) as product\_id

FROM orders

WHERE creation\_time::date between '2022-08-26'

and '2022-09-08') t2 using(order\_id) join products using(product\_id) join (SELECT to\_char(time, 'day') as weekday,

count(distinct user\_id) as all\_users

FROM user\_actions

WHERE time::date between '2022-08-26'

and '2022-09-08'

GROUP BY to\_char(time, 'day')) t3 using(weekday)

GROUP BY weekday, all\_users, weekday\_number

ORDER BY weekday\_number

Немного усложним наш первоначальный запрос и отдельно посчитаем ежедневную выручку с заказов новых пользователей нашего сервиса. Посмотрим, какую долю она составляет в общей выручке с заказов всех пользователей — и новых, и старых.

**Задание:**

Для каждого дня в таблицах orders и user\_actions рассчитайте следующие показатели:

1. Выручку, полученную в этот день.
2. Выручку с заказов новых пользователей, полученную в этот день.
3. Долю выручки с заказов новых пользователей в общей выручке, полученной за этот день.
4. Долю выручки с заказов остальных пользователей в общей выручке, полученной за этот день.

Колонки с показателями назовите соответственно revenue, new\_users\_revenue, new\_users\_revenue\_share, old\_users\_revenue\_share. Колонку с датами назовите date.

Все показатели долей необходимо выразить в процентах. При их расчёте округляйте значения до двух знаков после запятой.

Результат должен быть отсортирован по возрастанию даты.

Поля в результирующей таблице:

date, revenue, new\_users\_revenue, new\_users\_revenue\_share, old\_users\_revenue\_share

**Пояснение:**

Будем считать, что оплата за заказ поступает сразу же после его оформления, т.е. случаи, когда заказ был оформлен в один день, а оплата получена на следующий, возникнуть не могут.

Новыми будем считать тех пользователей, которые в данный день совершили своё первое действие в нашем сервисе.

with q as (SELECT date,

user\_id,

order\_id

FROM (SELECT time::date as date,

user\_id,

order\_id,

min(time::date) OVER (PARTITION BY user\_id) as first\_action

FROM user\_actions) t

WHERE date = first\_action

and order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')), w as (SELECT date,

sum(price) as new\_users\_revenue

FROM q join (SELECT unnest(product\_ids) as product\_id,

order\_id

FROM orders) t1 using(order\_id) join products using(product\_id)

GROUP BY date), e as (SELECT date,

sum(price) as revenue

FROM (SELECT creation\_time::date as date,

unnest(product\_ids) as product\_id

FROM orders

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')) t3 join products using(product\_id)

GROUP BY date)

SELECT date,

revenue,

new\_users\_revenue,

round(new\_users\_revenue::decimal / revenue \* 100, 2) as new\_users\_revenue\_share,

round((revenue - new\_users\_revenue)::decimal / revenue \* 100,

2) as old\_users\_revenue\_share

FROM w join e using(date)

ORDER BY date

Также было бы интересно посмотреть, какие товары пользуются наибольшим спросом и приносят нам основной доход.

**Задание:**

Для каждого товара, представленного в таблице products, за весь период времени в таблице orders рассчитайте следующие показатели:

1. Суммарную выручку, полученную от продажи этого товара за весь период.
2. Долю выручки от продажи этого товара в общей выручке, полученной за весь период.

Колонки с показателями назовите соответственно revenue и share\_in\_revenue. Колонку с наименованиями товаров назовите product\_name.

Долю выручки с каждого товара необходимо выразить в процентах. При её расчёте округляйте значения до двух знаков после запятой.

Товары, округлённая доля которых в выручке составляет менее 0.5%, объедините в общую группу с названием «ДРУГОЕ» (без кавычек), просуммировав округлённые доли этих товаров.

Результат должен быть отсортирован по убыванию выручки от продажи товара.

Поля в результирующей таблице: product\_name, revenue, share\_in\_revenue

**Пояснение:**

Будем считать, что оплата за заказ поступает сразу же после его оформления, т.е. случаи, когда заказ был оформлен в один день, а оплата получена на следующий, возникнуть не могут.

with q as (SELECT name as product\_name,

sum(price) as revenue

FROM products join (SELECT order\_id,

unnest(product\_ids) as product\_id

FROM orders

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')) t using(product\_id)

GROUP BY name, price

ORDER BY revenue desc), w as (SELECT product\_name,

revenue,

round(revenue::decimal / sum(revenue) OVER() \* 100, 2) as share\_in\_revenue

FROM q)

SELECT product\_name,

sum(revenue) as revenue,

sum(share\_in\_revenue) as share\_in\_revenue

FROM (SELECT case when product\_name in (SELECT product\_name

FROM w

WHERE share\_in\_revenue < 0.5) then 'ДРУГОЕ' else product\_name end as product\_name, revenue, share\_in\_revenue

FROM w) t

GROUP BY product\_name

ORDER BY revenue desc

Теперь попробуем учесть в наших расчётах затраты с налогами и посчитаем валовую прибыль, то есть ту сумму, которую мы фактически получили в результате реализации товаров за рассматриваемый период.

**Задание:**

Для каждого дня в таблицах orders и courier\_actions рассчитайте следующие показатели:

1. Выручку, полученную в этот день.
2. Затраты, образовавшиеся в этот день.
3. Сумму НДС с продажи товаров в этот день.
4. Валовую прибыль в этот день (выручка за вычетом затрат и НДС).
5. Суммарную выручку на текущий день.
6. Суммарные затраты на текущий день.
7. Суммарный НДС на текущий день.
8. Суммарную валовую прибыль на текущий день.
9. Долю валовой прибыли в выручке за этот день (долю п.4 в п.1).
10. Долю суммарной валовой прибыли в суммарной выручке на текущий день (долю п.8 в п.5).

Колонки с показателями назовите соответственно revenue, costs, tax, gross\_profit, total\_revenue, total\_costs, total\_tax, total\_gross\_profit, gross\_profit\_ratio, total\_gross\_profit\_ratio

Колонку с датами назовите date.

Долю валовой прибыли в выручке необходимо выразить в процентах, округлив значения до двух знаков после запятой.

Результат должен быть отсортирован по возрастанию даты.

Поля в результирующей таблице: date, revenue, costs, tax, gross\_profit, total\_revenue, total\_costs, total\_tax, total\_gross\_profit, gross\_profit\_ratio,total\_gross\_profit\_ratio

Чтобы посчитать затраты, в этой задаче введём дополнительные условия.

В упрощённом виде затраты нашего сервиса будем считать как сумму постоянных и переменных издержек. К постоянным издержкам отнесём аренду складских помещений, а к переменным — стоимость сборки и доставки заказа. Таким образом, переменные затраты будут напрямую зависеть от числа заказов.

Из данных, которые нам предоставил финансовый отдел, известно, что в августе 2022 года постоянные затраты составляли 120 000 рублей в день. Однако уже в сентябре нашему сервису потребовались дополнительные помещения, и поэтому постоянные затраты возросли до 150 000 рублей в день.

Также известно, что в августе 2022 года сборка одного заказа обходилась нам в 140 рублей, при этом курьерам мы платили по 150 рублей за один доставленный заказ и ещё 400 рублей ежедневно в качестве бонуса, если курьер доставлял не менее 5 заказов в день. В сентябре продакт-менеджерам удалось снизить затраты на сборку заказа до 115 рублей, но при этом пришлось повысить бонусную выплату за доставку 5 и более заказов до 500 рублей, чтобы обеспечить более конкурентоспособные условия труда. При этом в сентябре выплата курьерам за один доставленный заказ осталась неизменной.

**Пояснение:**

При расчёте переменных затрат учитывайте следующие условия:

1. Затраты на сборку учитываются в том же дне, когда был оформлен заказ. Сборка отменённых заказов не производится.

2. Выплата курьерам за доставленный заказ начисляется сразу же после его доставки, поэтому если курьер доставит заказ на следующий день, то и выплата будет учтена в следующем дне.

3. Для получения бонусной выплаты курьерам необходимо доставить не менее 5 заказов в течение одного дня, поэтому если курьер примет 5 заказов в течение дня, но последний из них доставит после полуночи, бонусную выплату он не получит.

При расчёте НДС учитывайте, что для некоторых товаров налог составляет 10%, а не 20%. Список товаров со сниженным НДС:

'сахар', 'сухарики', 'сушки', 'семечки',

'масло льняное', 'виноград', 'масло оливковое',

'арбуз', 'батон', 'йогурт', 'сливки', 'гречка',

'овсянка', 'макароны', 'баранина', 'апельсины',

'бублики', 'хлеб', 'горох', 'сметана', 'рыба копченая',

'мука', 'шпроты', 'сосиски', 'свинина', 'рис',

'масло кунжутное', 'сгущенка', 'ананас', 'говядина',

'соль', 'рыба вяленая', 'масло подсолнечное', 'яблоки',

'груши', 'лепешка', 'молоко', 'курица', 'лаваш', 'вафли', 'мандарины'

Также при расчёте величины НДС по каждому товару округляйте значения до двух знаков после запятой.

При расчёте выручки по-прежнему будем считать, что оплата за заказ поступает сразу же после его оформления, т.е. случаи, когда заказ был оформлен в один день, а оплата получена на следующий, возникнуть не могут.

with q as (SELECT DISTINCT time::date as date,

count(order\_id) filter(WHERE action = 'accept\_order') OVER(PARTITION BY time::date) as packing,

count(order\_id) filter(WHERE action = 'deliver\_order') OVER(PARTITION BY time::date) as delivered

FROM courier\_actions

WHERE order\_id in (SELECT order\_id

FROM courier\_actions

WHERE action = 'deliver\_order')), w as (SELECT date,

sum(bonus) as bonus

FROM (SELECT time::date as date,

count(distinct courier\_id) as bonus

FROM courier\_actions

WHERE action = 'deliver\_order'

GROUP BY time::date, courier\_id having count(distinct order\_id) >= 5) t

GROUP BY date), costs as (SELECT date,

case when date < '2022-09-01' then packing \* 140 + delivered \* 150 + coalesce(bonus,

0) \* 400 + 120000

when date >= '2022-09-01' then packing \* 115 + delivered \* 150 + coalesce(bonus,

0) \* 500 + 150000 end as costs

FROM q

LEFT JOIN w using(date)), revenue as (SELECT date,

sum(price) as revenue

FROM (SELECT creation\_time::date as date,

unnest(product\_ids) as product\_id,

order\_id

FROM orders) t1 join products using(product\_id)

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')

GROUP BY date), tax as (SELECT date,

sum(round(case when name in ('сахар', 'сухарики', 'сушки', 'семечки', 'масло льняное', 'виноград', 'масло оливковое', 'арбуз', 'батон', 'йогурт', 'сливки', 'гречка', 'овсянка', 'макароны', 'баранина', 'апельсины', 'бублики', 'хлеб', 'горох', 'сметана', 'рыба копченая', 'мука', 'шпроты', 'сосиски', 'свинина', 'рис', 'масло кунжутное', 'сгущенка', 'ананас', 'говядина', 'соль', 'рыба вяленая', 'масло подсолнечное', 'яблоки', 'груши', 'лепешка', 'молоко', 'курица', 'лаваш', 'вафли', 'мандарины') then price / 110 \* 10

else price / 120 \* 20 end, 2)) as tax

FROM (SELECT creation\_time::date as date,

unnest(product\_ids) as product\_id,

order\_id

FROM orders) t1 join products using(product\_id)

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')

GROUP BY date)

SELECT \*,

revenue - costs - tax as gross\_profit,

sum(revenue) OVER(ORDER BY date) as total\_revenue,

sum(costs) OVER(ORDER BY date) as total\_costs,

sum(tax) OVER(ORDER BY date) as total\_tax,

sum(revenue - costs - tax) OVER(ORDER BY date) as total\_gross\_profit,

round((revenue - costs - tax)::decimal / revenue \* 100, 2) as gross\_profit\_ratio,

round(sum(revenue - costs - tax) OVER(ORDER BY date)::decimal / sum(revenue) OVER(ORDER BY date) \* 100,

2) as total\_gross\_profit\_ratio

FROM revenue join costs using(date) join tax using(date)

Познакомимся с метриками, которые помогают оценить эффективность нашего маркетинга — то, насколько эффективно мы привлекаем пользователей в наше приложение.

Сперва рассмотрим метрику CAC (Customer Acquisition Cost), которая отражает затраты на привлечение одного покупателя.

Представим ситуацию: к нам обратились маркетологи с просьбой сравнить две рекламные кампании.

В рекламной кампании № 1 о нашем приложении рассказал известный блогер на Youtube-канале о кулинарии. На эту интеграцию суммарно потратили 250 тысяч рублей. В результате этой кампании 1 сентября в приложении зарегистрировался 171 человек.

В рамках рекламной кампании № 2 пользователям показывали таргетированную рекламу в социальных сетях. На это тоже суммарно потратили 250 тысяч рублей, и в результате 1 сентября у нас появилось 236 новых пользователей.

Как нам оценить, какой из каналов привлечения сработал лучше? На первый взгляд, вторая кампания показала себя лучше, поскольку нам удалось привлечь больше людей за те же деньги. Но не будем торопиться с выводами — давайте сначала проведём более подробный анализ и рассчитаем CAC для двух рекламных кампаний.

**Задание:**

На основе таблицы user\_actions рассчитайте метрику CAC для двух рекламных кампаний.

Список id пользователей, пришедших после рекламной кампании № 1:

8631, 8632, 8638, 8643, 8657, 8673, 8706, 8707, 8715, 8723, 8732, 8739, 8741,

8750, 8751, 8752, 8770, 8774, 8788, 8791, 8804, 8810, 8815, 8828, 8830, 8845,

8853, 8859, 8867, 8869, 8876, 8879, 8883, 8896, 8909, 8911, 8933, 8940, 8972,

8976, 8988, 8990, 9002, 9004, 9009, 9019, 9020, 9035, 9036, 9061, 9069, 9071,

9075, 9081, 9085, 9089, 9108, 9113, 9144, 9145, 9146, 9162, 9165, 9167, 9175,

9180, 9182, 9197, 9198, 9210, 9223, 9251, 9257, 9278, 9287, 9291, 9313, 9317,

9321, 9334, 9351, 9391, 9398, 9414, 9420, 9422, 9431, 9450, 9451, 9454, 9472,

9476, 9478, 9491, 9494, 9505, 9512, 9518, 9524, 9526, 9528, 9531, 9535, 9550,

9559, 9561, 9562, 9599, 9603, 9605, 9611, 9612, 9615, 9625, 9633, 9652, 9654,

9655, 9660, 9662, 9667, 9677, 9679, 9689, 9695, 9720, 9726, 9739, 9740, 9762,

9778, 9786, 9794, 9804, 9810, 9813, 9818, 9828, 9831, 9836, 9838, 9845, 9871,

9887, 9891, 9896, 9897, 9916, 9945, 9960, 9963, 9965, 9968, 9971, 9993, 9998,

9999, 10001, 10013, 10016, 10023, 10030, 10051, 10057, 10064, 10082, 10103,

10105, 10122, 10134, 10135

Список id пользователей, пришедших после рекламной кампании № 2:

8629, 8630, 8644, 8646, 8650, 8655, 8659, 8660, 8663, 8665, 8670, 8675, 8680, 8681,

8682, 8683, 8694, 8697, 8700, 8704, 8712, 8713, 8719, 8729, 8733, 8742, 8748, 8754,

8771, 8794, 8795, 8798, 8803, 8805, 8806, 8812, 8814, 8825, 8827, 8838, 8849, 8851,

8854, 8855, 8870, 8878, 8882, 8886, 8890, 8893, 8900, 8902, 8913, 8916, 8923, 8929,

8935, 8942, 8943, 8949, 8953, 8955, 8966, 8968, 8971, 8973, 8980, 8995, 8999, 9000,

9007, 9013, 9041, 9042, 9047, 9064, 9068, 9077, 9082, 9083, 9095, 9103, 9109, 9117,

9123, 9127, 9131, 9137, 9140, 9149, 9161, 9179, 9181, 9183, 9185, 9190, 9196, 9203,

9207, 9226, 9227, 9229, 9230, 9231, 9250, 9255, 9259, 9267, 9273, 9281, 9282, 9289,

9292, 9303, 9310, 9312, 9315, 9327, 9333, 9335, 9337, 9343, 9356, 9368, 9370, 9383,

9392, 9404, 9410, 9421, 9428, 9432, 9437, 9468, 9479, 9483, 9485, 9492, 9495, 9497,

9498, 9500, 9510, 9527, 9529, 9530, 9538, 9539, 9545, 9557, 9558, 9560, 9564, 9567,

9570, 9591, 9596, 9598, 9616, 9631, 9634, 9635, 9636, 9658, 9666, 9672, 9684, 9692,

9700, 9704, 9706, 9711, 9719, 9727, 9735, 9741, 9744, 9749, 9752, 9753, 9755, 9757,

9764, 9783, 9784, 9788, 9790, 9808, 9820, 9839, 9841, 9843, 9853, 9855, 9859, 9863,

9877, 9879, 9880, 9882, 9883, 9885, 9901, 9904, 9908, 9910, 9912, 9920, 9929, 9930,

9935, 9939, 9958, 9959, 9961, 9983, 10027, 10033, 10038, 10045, 10047, 10048, 10058,

10059, 10067, 10069, 10073, 10075, 10078, 10079, 10081, 10092, 10106, 10110, 10113, 10131

Колонку с наименованиями кампаний назовите ads\_campaign, а колонку со значением метрики — cac.

Наименования кампаний выведите в следующем виде:

Кампания № 1

Кампания № 2

Полученные значения метрики округлите до двух знаков после запятой.

Результат должен быть отсортирован по убыванию значения метрики.

Поля в результирующей таблице: ads\_campaign, cac

**Пояснение:**

Покупателями будем считать тех пользователей, которые сделали хотя бы один заказ, который в дальнейшем не был отменён. Например, если человек сделал только один заказ, а потом отменил его, то покупателем мы его не считаем.

with q as (SELECT \*,

case when user\_id in (8631, 8632, 8638, 8643, 8657, 8673, 8706, 8707, 8715, 8723, 8732,

8739, 8741, 8750, 8751, 8752, 8770, 8774, 8788, 8791,

8804, 8810, 8815, 8828, 8830, 8845, 8853, 8859, 8867,

8869, 8876, 8879, 8883, 8896, 8909, 8911, 8933, 8940,

8972, 8976, 8988, 8990, 9002, 9004, 9009, 9019, 9020,

9035, 9036, 9061, 9069, 9071, 9075, 9081, 9085, 9089,

9108, 9113, 9144, 9145, 9146, 9162, 9165, 9167, 9175,

9180, 9182, 9197, 9198, 9210, 9223, 9251, 9257, 9278,

9287, 9291, 9313, 9317, 9321, 9334, 9351, 9391, 9398,

9414, 9420, 9422, 9431, 9450, 9451, 9454, 9472, 9476,

9478, 9491, 9494, 9505, 9512, 9518, 9524, 9526, 9528,

9531, 9535, 9550, 9559, 9561, 9562, 9599, 9603, 9605,

9611, 9612, 9615, 9625, 9633, 9652, 9654, 9655, 9660,

9662, 9667, 9677, 9679, 9689, 9695, 9720, 9726, 9739,

9740, 9762, 9778, 9786, 9794, 9804, 9810, 9813, 9818,

9828, 9831, 9836, 9838, 9845, 9871, 9887, 9891, 9896,

9897, 9916, 9945, 9960, 9963, 9965, 9968, 9971, 9993,

9998, 9999, 10001, 10013, 10016, 10023, 10030, 10051,

10057, 10064, 10082, 10103, 10105, 10122, 10134, 10135) then 'Кампания № 1'

when user\_id in (8629, 8630, 8644, 8646, 8650, 8655, 8659, 8660, 8663, 8665, 8670,

8675, 8680, 8681, 8682, 8683, 8694, 8697, 8700, 8704,

8712, 8713, 8719, 8729, 8733, 8742, 8748, 8754, 8771,

8794, 8795, 8798, 8803, 8805, 8806, 8812, 8814, 8825,

8827, 8838, 8849, 8851, 8854, 8855, 8870, 8878, 8882,

8886, 8890, 8893, 8900, 8902, 8913, 8916, 8923, 8929,

8935, 8942, 8943, 8949, 8953, 8955, 8966, 8968, 8971,

8973, 8980, 8995, 8999, 9000, 9007, 9013, 9041, 9042,

9047, 9064, 9068, 9077, 9082, 9083, 9095, 9103, 9109,

9117, 9123, 9127, 9131, 9137, 9140, 9149, 9161, 9179,

9181, 9183, 9185, 9190, 9196, 9203, 9207, 9226, 9227,

9229, 9230, 9231, 9250, 9255, 9259, 9267, 9273, 9281,

9282, 9289, 9292, 9303, 9310, 9312, 9315, 9327, 9333,

9335, 9337, 9343, 9356, 9368, 9370, 9383, 9392, 9404,

9410, 9421, 9428, 9432, 9437, 9468, 9479, 9483, 9485,

9492, 9495, 9497, 9498, 9500, 9510, 9527, 9529, 9530,

9538, 9539, 9545, 9557, 9558, 9560, 9564, 9567, 9570,

9591, 9596, 9598, 9616, 9631, 9634, 9635, 9636, 9658,

9666, 9672, 9684, 9692, 9700, 9704, 9706, 9711, 9719,

9727, 9735, 9741, 9744, 9749, 9752, 9753, 9755, 9757,

9764, 9783, 9784, 9788, 9790, 9808, 9820, 9839, 9841,

9843, 9853, 9855, 9859, 9863, 9877, 9879, 9880, 9882,

9883, 9885, 9901, 9904, 9908, 9910, 9912, 9920, 9929,

9930, 9935, 9939, 9958, 9959, 9961, 9983, 10027, 10033,

10038, 10045, 10047, 10048, 10058, 10059, 10067, 10069,

10073, 10075, 10078, 10079, 10081, 10092, 10106, 10110,

10113, 10131) then 'Кампания № 2' end as ads\_campaign

FROM user\_actions

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order'))

SELECT ads\_campaign,

round(250000::decimal / count(distinct user\_id), 2) as cac

FROM q

WHERE ads\_campaign is not null

GROUP BY ads\_campaign

Отлично, CAC мы рассчитали, но можем ли мы теперь утверждать, что вторая рекламная кампания лучше только потому, что позволяет привлекать платящих пользователей за меньшие деньги?

Нет, делать такие выводы рано — давайте оценим ещё один показатель, который отражает рентабельность инвестиций и показывает, насколько выгодным является тот или иной проект или продукт. Эта метрика называется ROI (Return on Investment), в маркетинге её часто применяют для подсчёта окупаемости рекламных кампаний.

Приведём пример. Допустим, мы вложили в рекламу 100 рублей, и в результате продали товаров на 220 рублей. Тогда ROI будет рассчитан следующим образом:

*ROI*=(220−100)/100∗100%=120%

Таким образом, на каждый вложенный в рекламу рубль мы получили 1,2 рубля дохода.

Давайте сравним две рекламные кампании по значениям метрики ROI и сделаем вывод о том, какой из рекламных каналов в большей мере окупает расходы на привлечение новых пользователей.

**Задание:**

Рассчитайте ROI для каждого рекламного канала.

Колонку с наименованиями кампаний назовите ads\_campaign, а колонку со значением метрики — roi.

Наименования кампаний выведите в следующем виде:

Кампания № 1

Кампания № 2

Полученные значения метрики необходимо выразить в процентах и округлить до двух знаков после запятой.

Результат должен быть отсортирован по убыванию значения метрики.

Поля в результирующей таблице: ads\_campaign, roi

**Пояснение:**

Покупателями будем считать тех пользователей, которые сделали хотя бы один заказ, который в дальнейшем не был отменён. Например, если человек сделал только один заказ, а потом отменил его, то покупателем мы его не считаем.

Не забывайте при делении заранее приводить значения к нужному типу данных.

Списки зарегистрировавшихся пользователей те же, что и на предыдущем шаге.

with q as (SELECT \*,

case when user\_id in (8631, 8632, 8638, 8643, 8657, 8673, 8706, 8707, 8715, 8723, 8732,

8739, 8741, 8750, 8751, 8752, 8770, 8774, 8788, 8791,

8804, 8810, 8815, 8828, 8830, 8845, 8853, 8859, 8867,

8869, 8876, 8879, 8883, 8896, 8909, 8911, 8933, 8940,

8972, 8976, 8988, 8990, 9002, 9004, 9009, 9019, 9020,

9035, 9036, 9061, 9069, 9071, 9075, 9081, 9085, 9089,

9108, 9113, 9144, 9145, 9146, 9162, 9165, 9167, 9175,

9180, 9182, 9197, 9198, 9210, 9223, 9251, 9257, 9278,

9287, 9291, 9313, 9317, 9321, 9334, 9351, 9391, 9398,

9414, 9420, 9422, 9431, 9450, 9451, 9454, 9472, 9476,

9478, 9491, 9494, 9505, 9512, 9518, 9524, 9526, 9528,

9531, 9535, 9550, 9559, 9561, 9562, 9599, 9603, 9605,

9611, 9612, 9615, 9625, 9633, 9652, 9654, 9655, 9660,

9662, 9667, 9677, 9679, 9689, 9695, 9720, 9726, 9739,

9740, 9762, 9778, 9786, 9794, 9804, 9810, 9813, 9818,

9828, 9831, 9836, 9838, 9845, 9871, 9887, 9891, 9896,

9897, 9916, 9945, 9960, 9963, 9965, 9968, 9971, 9993,

9998, 9999, 10001, 10013, 10016, 10023, 10030, 10051,

10057, 10064, 10082, 10103, 10105, 10122, 10134, 10135) then 'Кампания № 1'

when user\_id in (8629, 8630, 8644, 8646, 8650, 8655, 8659, 8660, 8663, 8665, 8670,

8675, 8680, 8681, 8682, 8683, 8694, 8697, 8700, 8704,

8712, 8713, 8719, 8729, 8733, 8742, 8748, 8754, 8771,

8794, 8795, 8798, 8803, 8805, 8806, 8812, 8814, 8825,

8827, 8838, 8849, 8851, 8854, 8855, 8870, 8878, 8882,

8886, 8890, 8893, 8900, 8902, 8913, 8916, 8923, 8929,

8935, 8942, 8943, 8949, 8953, 8955, 8966, 8968, 8971,

8973, 8980, 8995, 8999, 9000, 9007, 9013, 9041, 9042,

9047, 9064, 9068, 9077, 9082, 9083, 9095, 9103, 9109,

9117, 9123, 9127, 9131, 9137, 9140, 9149, 9161, 9179,

9181, 9183, 9185, 9190, 9196, 9203, 9207, 9226, 9227,

9229, 9230, 9231, 9250, 9255, 9259, 9267, 9273, 9281,

9282, 9289, 9292, 9303, 9310, 9312, 9315, 9327, 9333,

9335, 9337, 9343, 9356, 9368, 9370, 9383, 9392, 9404,

9410, 9421, 9428, 9432, 9437, 9468, 9479, 9483, 9485,

9492, 9495, 9497, 9498, 9500, 9510, 9527, 9529, 9530,

9538, 9539, 9545, 9557, 9558, 9560, 9564, 9567, 9570,

9591, 9596, 9598, 9616, 9631, 9634, 9635, 9636, 9658,

9666, 9672, 9684, 9692, 9700, 9704, 9706, 9711, 9719,

9727, 9735, 9741, 9744, 9749, 9752, 9753, 9755, 9757,

9764, 9783, 9784, 9788, 9790, 9808, 9820, 9839, 9841,

9843, 9853, 9855, 9859, 9863, 9877, 9879, 9880, 9882,

9883, 9885, 9901, 9904, 9908, 9910, 9912, 9920, 9929,

9930, 9935, 9939, 9958, 9959, 9961, 9983, 10027, 10033,

10038, 10045, 10047, 10048, 10058, 10059, 10067, 10069,

10073, 10075, 10078, 10079, 10081, 10092, 10106, 10110,

10113, 10131) then 'Кампания № 2' end as ads\_campaign

FROM user\_actions

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order'))

SELECT ads\_campaign,

round((sum(price) - 250000) / 250000 \* 100, 2) as roi

FROM q join (SELECT order\_id,

unnest(product\_ids) as product\_id

FROM orders) t using(order\_id) join products using(product\_id)

WHERE ads\_campaign is not null

GROUP BY ads\_campaign

ORDER BY roi desc

**Задание:**

Для каждой рекламной кампании посчитайте среднюю стоимость заказа привлечённых пользователей за первую неделю использования приложения с 1 по 7 сентября 2022 года.

Колонку с наименованиями кампаний назовите ads\_campaign, а колонку со значением метрики — avg\_check.

Наименования кампаний выведите в следующем виде:

Кампания № 1

Кампания № 2

Полученные значения метрики необходимо округлить до двух знаков после запятой.

Результат должен быть отсортирован по убыванию значения метрики.

Поля в результирующей таблице: ads\_campaign, avg\_check

**Пояснение:**

Покупателями будем считать тех пользователей, которые сделали хотя бы один заказ, который в дальнейшем не был отменён. Например, если человек сделал только один заказ, а потом отменил его, то покупателем мы его не считаем.

Не забывайте при делении заранее приводить значения к нужному типу данных.

Списки зарегистрировавшихся пользователей те же, что и на предыдущих шагах.

with q as (SELECT \*,

case when user\_id in (8631, 8632, 8638, 8643, 8657, 8673, 8706, 8707, 8715, 8723, 8732,

8739, 8741, 8750, 8751, 8752, 8770, 8774, 8788, 8791,

8804, 8810, 8815, 8828, 8830, 8845, 8853, 8859, 8867,

8869, 8876, 8879, 8883, 8896, 8909, 8911, 8933, 8940,

8972, 8976, 8988, 8990, 9002, 9004, 9009, 9019, 9020,

9035, 9036, 9061, 9069, 9071, 9075, 9081, 9085, 9089,

9108, 9113, 9144, 9145, 9146, 9162, 9165, 9167, 9175,

9180, 9182, 9197, 9198, 9210, 9223, 9251, 9257, 9278,

9287, 9291, 9313, 9317, 9321, 9334, 9351, 9391, 9398,

9414, 9420, 9422, 9431, 9450, 9451, 9454, 9472, 9476,

9478, 9491, 9494, 9505, 9512, 9518, 9524, 9526, 9528,

9531, 9535, 9550, 9559, 9561, 9562, 9599, 9603, 9605,

9611, 9612, 9615, 9625, 9633, 9652, 9654, 9655, 9660,

9662, 9667, 9677, 9679, 9689, 9695, 9720, 9726, 9739,

9740, 9762, 9778, 9786, 9794, 9804, 9810, 9813, 9818,

9828, 9831, 9836, 9838, 9845, 9871, 9887, 9891, 9896,

9897, 9916, 9945, 9960, 9963, 9965, 9968, 9971, 9993,

9998, 9999, 10001, 10013, 10016, 10023, 10030, 10051,

10057, 10064, 10082, 10103, 10105, 10122, 10134, 10135) then 'Кампания № 1'

when user\_id in (8629, 8630, 8644, 8646, 8650, 8655, 8659, 8660, 8663, 8665, 8670,

8675, 8680, 8681, 8682, 8683, 8694, 8697, 8700, 8704,

8712, 8713, 8719, 8729, 8733, 8742, 8748, 8754, 8771,

8794, 8795, 8798, 8803, 8805, 8806, 8812, 8814, 8825,

8827, 8838, 8849, 8851, 8854, 8855, 8870, 8878, 8882,

8886, 8890, 8893, 8900, 8902, 8913, 8916, 8923, 8929,

8935, 8942, 8943, 8949, 8953, 8955, 8966, 8968, 8971,

8973, 8980, 8995, 8999, 9000, 9007, 9013, 9041, 9042,

9047, 9064, 9068, 9077, 9082, 9083, 9095, 9103, 9109,

9117, 9123, 9127, 9131, 9137, 9140, 9149, 9161, 9179,

9181, 9183, 9185, 9190, 9196, 9203, 9207, 9226, 9227,

9229, 9230, 9231, 9250, 9255, 9259, 9267, 9273, 9281,

9282, 9289, 9292, 9303, 9310, 9312, 9315, 9327, 9333,

9335, 9337, 9343, 9356, 9368, 9370, 9383, 9392, 9404,

9410, 9421, 9428, 9432, 9437, 9468, 9479, 9483, 9485,

9492, 9495, 9497, 9498, 9500, 9510, 9527, 9529, 9530,

9538, 9539, 9545, 9557, 9558, 9560, 9564, 9567, 9570,

9591, 9596, 9598, 9616, 9631, 9634, 9635, 9636, 9658,

9666, 9672, 9684, 9692, 9700, 9704, 9706, 9711, 9719,

9727, 9735, 9741, 9744, 9749, 9752, 9753, 9755, 9757,

9764, 9783, 9784, 9788, 9790, 9808, 9820, 9839, 9841,

9843, 9853, 9855, 9859, 9863, 9877, 9879, 9880, 9882,

9883, 9885, 9901, 9904, 9908, 9910, 9912, 9920, 9929,

9930, 9935, 9939, 9958, 9959, 9961, 9983, 10027, 10033,

10038, 10045, 10047, 10048, 10058, 10059, 10067, 10069,

10073, 10075, 10078, 10079, 10081, 10092, 10106, 10110,

10113, 10131) then 'Кампания № 2' end as ads\_campaign

FROM user\_actions

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order'))

SELECT ads\_campaign,

round(avg(avg), 2) as avg\_check

FROM (SELECT ads\_campaign,

user\_id,

avg(sum)

FROM (SELECT ads\_campaign,

user\_id,

order\_id,

sum(price)

FROM q join (SELECT order\_id,

unnest(product\_ids) as product\_id

FROM orders) t using(order\_id) join products using(product\_id)

WHERE ads\_campaign is not null

and time::date between '2022-09-01'

and '2022-09-07'

GROUP BY user\_id, order\_id, ads\_campaign) t

GROUP BY ads\_campaign, user\_id) t1

GROUP BY ads\_campaign

ORDER BY avg\_check desc

Retention

Средний чек мы посчитали, но ответа на наш вопрос не получили. В чём же тогда может быть дело? Давайте обратим внимание на ещё один важный показатель — Retention rate.

Retention rate — коэффициент удержания клиентов. Он показывает долю пользователей, которые вернулись в приложение спустя N дней, недель или месяцев после своего первого входа — метрику можно рассчитывать по любым периодам, которые интересуют.

В случае нашего сервиса доставки высокий Retention означает, что пользователи часто возвращаются, чтобы сделать повторный заказ. Низкий Retention, напротив, говорит о том, что большинство пользователей уходят в отток после одного или нескольких взаимодействий. Возможно, им неудобно пользоваться нашим приложением, или же их не устраивают цены, ассортимент товаров, скорость или стоимость доставки — причины могут быть самые разные, и чтобы их установить, нужны дополнительные исследования.

Таким образом, метрика Retention отражает ценность нашего продукта в глазах пользователей: если они возвращаются к нам с определённой периодичностью, значит, наш продукт помогает им решать свои задачи.

Возможно, именно этот показатель поможет нам ответить на вопрос, почему одна из двух рекламных кампаний показывает себя лучше. Но прежде чем приступать к анализу Retention, давайте подробнее разберёмся с методикой его расчёта.

На основе данных в таблице user\_actions рассчитайте показатель дневного Retention для всех пользователей, разбив их на когорты по дате первого взаимодействия с нашим приложением.

В результат включите четыре колонки: месяц первого взаимодействия, дату первого взаимодействия, количество дней, прошедших с даты первого взаимодействия (порядковый номер дня начиная с 0), и само значение Retention.

Колонки со значениями назовите соответственно start\_month, start\_date, day\_number, retention.

Метрику необходимо выразить в виде доли, округлив полученные значения до двух знаков после запятой.

Месяц первого взаимодействия укажите в виде даты, округлённой до первого числа месяца.

Результат должен быть отсортирован сначала по возрастанию даты первого взаимодействия, затем по возрастанию порядкового номера дня.

Поля в результирующей таблице: start\_month, start\_date, day\_number, retention

with q as (SELECT time::date as dt,

user\_id,

min(time::date) OVER (PARTITION BY user\_id) as start\_date

FROM user\_actions)

SELECT date\_trunc('month', start\_date)::date as start\_month,

start\_date,

dt - start\_date as day\_number,

round(count(distinct user\_id)::decimal / max(count(distinct user\_id)) OVER(PARTITION BY start\_date),

2) as retention

FROM q

GROUP BY dt, start\_date, date\_trunc('month', dt)

ORDER BY start\_date

А теперь вернёмся к анализу рекламных кампаний и посчитаем Retention для двух групп пользователей. Возможно, дело именно в том, что один из каналов привёл более активных и заинтересованных пользователей. Давайте это выясним!

**Задание:**

Для каждой рекламной кампании посчитайте Retention 1-го и 7-го дня у привлечённых пользователей.

В результат включите четыре колонки: колонку с наименованиями кампаний, дату первого взаимодействия с приложением, количество дней, прошедших с даты первого взаимодействия (порядковый номер), и само значение Retention.

Колонки со значениями назовите соответственно ads\_campaign, start\_date, day\_number, retention.

Наименования кампаний выведите в следующем виде:

Кампания № 1

Кампания № 2

Метрику необходимо выразить в виде доли, округлив полученные значения до двух знаков после запятой.

Результат должен быть отсортирован сначала по наименованию кампании (по возрастанию), затем по возрастанию порядкового номера дня.

В результат должны попасть следующие дни: нулевой, первый и седьмой.

Поля в результирующей таблице: ads\_campaign, start\_date, day\_number, retention

with q as (SELECT \*,

case when user\_id in (8631, 8632, 8638, 8643, 8657, 8673, 8706, 8707, 8715, 8723, 8732,

8739, 8741, 8750, 8751, 8752, 8770, 8774, 8788, 8791,

8804, 8810, 8815, 8828, 8830, 8845, 8853, 8859, 8867,

8869, 8876, 8879, 8883, 8896, 8909, 8911, 8933, 8940,

8972, 8976, 8988, 8990, 9002, 9004, 9009, 9019, 9020,

9035, 9036, 9061, 9069, 9071, 9075, 9081, 9085, 9089,

9108, 9113, 9144, 9145, 9146, 9162, 9165, 9167, 9175,

9180, 9182, 9197, 9198, 9210, 9223, 9251, 9257, 9278,

9287, 9291, 9313, 9317, 9321, 9334, 9351, 9391, 9398,

9414, 9420, 9422, 9431, 9450, 9451, 9454, 9472, 9476,

9478, 9491, 9494, 9505, 9512, 9518, 9524, 9526, 9528,

9531, 9535, 9550, 9559, 9561, 9562, 9599, 9603, 9605,

9611, 9612, 9615, 9625, 9633, 9652, 9654, 9655, 9660,

9662, 9667, 9677, 9679, 9689, 9695, 9720, 9726, 9739,

9740, 9762, 9778, 9786, 9794, 9804, 9810, 9813, 9818,

9828, 9831, 9836, 9838, 9845, 9871, 9887, 9891, 9896,

9897, 9916, 9945, 9960, 9963, 9965, 9968, 9971, 9993,

9998, 9999, 10001, 10013, 10016, 10023, 10030, 10051,

10057, 10064, 10082, 10103, 10105, 10122, 10134, 10135) then 'Кампания № 1'

when user\_id in (8629, 8630, 8644, 8646, 8650, 8655, 8659, 8660, 8663, 8665, 8670,

8675, 8680, 8681, 8682, 8683, 8694, 8697, 8700, 8704,

8712, 8713, 8719, 8729, 8733, 8742, 8748, 8754, 8771,

8794, 8795, 8798, 8803, 8805, 8806, 8812, 8814, 8825,

8827, 8838, 8849, 8851, 8854, 8855, 8870, 8878, 8882,

8886, 8890, 8893, 8900, 8902, 8913, 8916, 8923, 8929,

8935, 8942, 8943, 8949, 8953, 8955, 8966, 8968, 8971,

8973, 8980, 8995, 8999, 9000, 9007, 9013, 9041, 9042,

9047, 9064, 9068, 9077, 9082, 9083, 9095, 9103, 9109,

9117, 9123, 9127, 9131, 9137, 9140, 9149, 9161, 9179,

9181, 9183, 9185, 9190, 9196, 9203, 9207, 9226, 9227,

9229, 9230, 9231, 9250, 9255, 9259, 9267, 9273, 9281,

9282, 9289, 9292, 9303, 9310, 9312, 9315, 9327, 9333,

9335, 9337, 9343, 9356, 9368, 9370, 9383, 9392, 9404,

9410, 9421, 9428, 9432, 9437, 9468, 9479, 9483, 9485,

9492, 9495, 9497, 9498, 9500, 9510, 9527, 9529, 9530,

9538, 9539, 9545, 9557, 9558, 9560, 9564, 9567, 9570,

9591, 9596, 9598, 9616, 9631, 9634, 9635, 9636, 9658,

9666, 9672, 9684, 9692, 9700, 9704, 9706, 9711, 9719,

9727, 9735, 9741, 9744, 9749, 9752, 9753, 9755, 9757,

9764, 9783, 9784, 9788, 9790, 9808, 9820, 9839, 9841,

9843, 9853, 9855, 9859, 9863, 9877, 9879, 9880, 9882,

9883, 9885, 9901, 9904, 9908, 9910, 9912, 9920, 9929,

9930, 9935, 9939, 9958, 9959, 9961, 9983, 10027, 10033,

10038, 10045, 10047, 10048, 10058, 10059, 10067, 10069,

10073, 10075, 10078, 10079, 10081, 10092, 10106, 10110,

10113, 10131) then 'Кампания № 2' end as ads\_campaign

FROM user\_actions), w as (SELECT ads\_campaign,

time::date as dt,

user\_id,

min(time::date) OVER(PARTITION BY user\_id) as start\_date,

time::date - min(time::date) OVER(PARTITION BY user\_id) as day\_number

FROM q

WHERE ads\_campaign is not null)

SELECT ads\_campaign,

start\_date,

day\_number,

round(count(distinct user\_id)::decimal / max(count(distinct user\_id)) OVER(PARTITION BY ads\_campaign,

start\_date), 2) as retention

FROM w

WHERE day\_number in (0, 1, 7)

GROUP BY ads\_campaign, start\_date, day\_number

ORDER BY ads\_campaign, day\_number

И напоследок давайте выясним, на какой день доход от заказов покупателей, пришедших после первой рекламной кампании, превысил расходы на их привлечение. Для этого рассчитаем накопительный ARPPU и сравним его с уже посчитанной ранее метрикой CAC. Сделаем это для двух кампаний, чтобы точно убедиться, что в случае второй рекламной кампании затраты мы пока не окупили.

Суть расчёта накопительного ARPPU заключается в том, что для каждого дня число покупателей в когорте будет оставаться неизменным, а полученный от их заказов доход — расти. В результате накопительный ARPPU будет постепенно увеличиваться и в какой-то момент превысит сумму изначальных затрат на привлечение одного покупателя.

**Задание:**

Для каждой рекламной кампании для каждого дня посчитайте две метрики:

1. Накопительный ARPPU.

2. Затраты на привлечение одного покупателя (CAC).

Колонку с наименованиями кампаний назовите ads\_campaign, колонку с днями — day, а колонки со значениями метрик — cumulative\_arppu и cac.

Значения метрики CAC укажите одинаковым для всех дней (это необходимо для наглядной визуализации).

Наименования кампаний выведите в следующем виде:

Кампания № 1

Кампания № 2

Дни пронумеруйте начиная с 0 и отобразите в следующем формате:

Day 0

Day 1

Day 2

и т.д.

Полученные значения метрик необходимо округлить до двух знаков после запятой.

Результат должен быть отсортирован сначала по наименованию кампании (по возрастанию), затем по наименованию дня (также по возрастанию).

Поля в результирующей таблице: ads\_campaign, day, cumulative\_arppu, cac

**Пояснение:**

Покупателями будем считать тех пользователей, которые сделали хотя бы один заказ, который в дальнейшем не был отменён. Например, если человек сделал только один заказ, а потом отменил его, то покупателем мы его не считаем.

with q as (SELECT \*,

case when user\_id in (8631, 8632, 8638, 8643, 8657, 8673, 8706, 8707, 8715, 8723, 8732,

8739, 8741, 8750, 8751, 8752, 8770, 8774, 8788, 8791,

8804, 8810, 8815, 8828, 8830, 8845, 8853, 8859, 8867,

8869, 8876, 8879, 8883, 8896, 8909, 8911, 8933, 8940,

8972, 8976, 8988, 8990, 9002, 9004, 9009, 9019, 9020,

9035, 9036, 9061, 9069, 9071, 9075, 9081, 9085, 9089,

9108, 9113, 9144, 9145, 9146, 9162, 9165, 9167, 9175,

9180, 9182, 9197, 9198, 9210, 9223, 9251, 9257, 9278,

9287, 9291, 9313, 9317, 9321, 9334, 9351, 9391, 9398,

9414, 9420, 9422, 9431, 9450, 9451, 9454, 9472, 9476,

9478, 9491, 9494, 9505, 9512, 9518, 9524, 9526, 9528,

9531, 9535, 9550, 9559, 9561, 9562, 9599, 9603, 9605,

9611, 9612, 9615, 9625, 9633, 9652, 9654, 9655, 9660,

9662, 9667, 9677, 9679, 9689, 9695, 9720, 9726, 9739,

9740, 9762, 9778, 9786, 9794, 9804, 9810, 9813, 9818,

9828, 9831, 9836, 9838, 9845, 9871, 9887, 9891, 9896,

9897, 9916, 9945, 9960, 9963, 9965, 9968, 9971, 9993,

9998, 9999, 10001, 10013, 10016, 10023, 10030, 10051,

10057, 10064, 10082, 10103, 10105, 10122, 10134, 10135) then 'Кампания № 1'

when user\_id in (8629, 8630, 8644, 8646, 8650, 8655, 8659, 8660, 8663, 8665, 8670,

8675, 8680, 8681, 8682, 8683, 8694, 8697, 8700, 8704,

8712, 8713, 8719, 8729, 8733, 8742, 8748, 8754, 8771,

8794, 8795, 8798, 8803, 8805, 8806, 8812, 8814, 8825,

8827, 8838, 8849, 8851, 8854, 8855, 8870, 8878, 8882,

8886, 8890, 8893, 8900, 8902, 8913, 8916, 8923, 8929,

8935, 8942, 8943, 8949, 8953, 8955, 8966, 8968, 8971,

8973, 8980, 8995, 8999, 9000, 9007, 9013, 9041, 9042,

9047, 9064, 9068, 9077, 9082, 9083, 9095, 9103, 9109,

9117, 9123, 9127, 9131, 9137, 9140, 9149, 9161, 9179,

9181, 9183, 9185, 9190, 9196, 9203, 9207, 9226, 9227,

9229, 9230, 9231, 9250, 9255, 9259, 9267, 9273, 9281,

9282, 9289, 9292, 9303, 9310, 9312, 9315, 9327, 9333,

9335, 9337, 9343, 9356, 9368, 9370, 9383, 9392, 9404,

9410, 9421, 9428, 9432, 9437, 9468, 9479, 9483, 9485,

9492, 9495, 9497, 9498, 9500, 9510, 9527, 9529, 9530,

9538, 9539, 9545, 9557, 9558, 9560, 9564, 9567, 9570,

9591, 9596, 9598, 9616, 9631, 9634, 9635, 9636, 9658,

9666, 9672, 9684, 9692, 9700, 9704, 9706, 9711, 9719,

9727, 9735, 9741, 9744, 9749, 9752, 9753, 9755, 9757,

9764, 9783, 9784, 9788, 9790, 9808, 9820, 9839, 9841,

9843, 9853, 9855, 9859, 9863, 9877, 9879, 9880, 9882,

9883, 9885, 9901, 9904, 9908, 9910, 9912, 9920, 9929,

9930, 9935, 9939, 9958, 9959, 9961, 9983, 10027, 10033,

10038, 10045, 10047, 10048, 10058, 10059, 10067, 10069,

10073, 10075, 10078, 10079, 10081, 10092, 10106, 10110,

10113, 10131) then 'Кампания № 2' end as ads\_campaign

FROM user\_actions)

SELECT ads\_campaign,

day,

round(sum(arppu) OVER(PARTITION BY ads\_campaign

ORDER BY day), 2) as cumulative\_arppu,

cac

FROM (SELECT ads\_campaign,

concat('Day ', time::date - min(time::date) OVER()) as day,

sum(price) filter(WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')) / paying\_users as arppu, round(250000::decimal / paying\_users, 2) as cac

FROM q join (SELECT order\_id,

unnest(product\_ids) as product\_id

FROM orders) t using(order\_id) join products using(product\_id) join (SELECT ads\_campaign,

count(distinct user\_id) as paying\_users

FROM q

WHERE order\_id not in (SELECT order\_id

FROM user\_actions

WHERE action = 'cancel\_order')

GROUP BY ads\_campaign) t1 using(ads\_campaign)

WHERE ads\_campaign is not null

GROUP BY ads\_campaign, time::date, paying\_users) t2